

How Reference Architectures support the evolution of Product Families; the Darwin research project

by *Gerrit Muller* Embedded Systems Institute
e-mail: `gerrit.muller@embeddedsystems.nl`
`www.gaudisite.nl`

Abstract

TBD

Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

This work has been carried out as part of the Darwin project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the BSIK program.

February 10, 2011
status: planned
version: 0.4

logo
TBD

High Level Problem Statement

Installed Base Business
Life Cycle Management

costly
high effort

*diversity and # of
configurations*

Development efficiency

costly
high effort
too late

Innovation rate

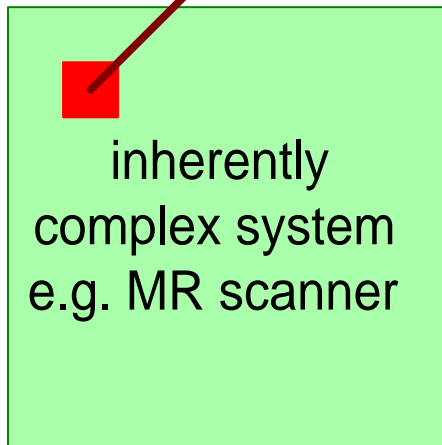
too low
too late



see next
slides

The Innovation Challenge

potential
innovation:
change



inherently
complex system
e.g. MR scanner

Challenge:

how to apply change locally for exploration of potential value and feasibility?

Postulate 1:

for effective exploration the following properties must be maintained

patient throughput
system responsiveness
image quality
safety
reliability

Postulate 2:

a system architecture that supports this level of exploration also supports the next phases of innovation: scaling-up and engineering

Postulate 3:

a system architecture that supports this level of exploration also supports life cycle business over many generations

Evolvability Problem Statement

exploration is difficult

too much
time, effort, cost

from idea to tryout

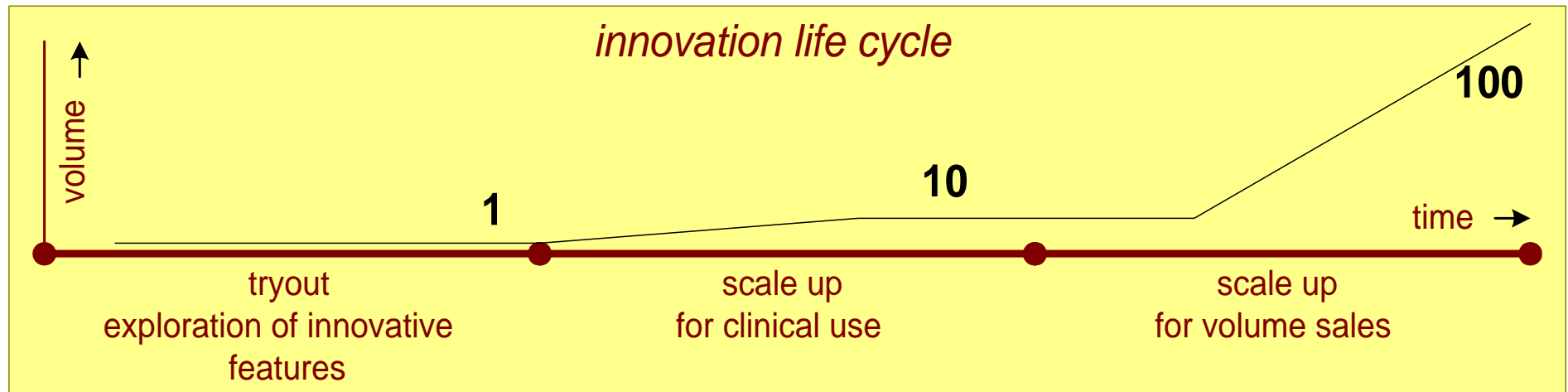
reliable realization is difficult

too much
and unpredictable
development
time, effort, cost

from tryout to realization

engineering is difficult

some new features
late relative to competition
too much
material and labor cost



Evolvability Problem Analysis

problems

exploration is difficult
too much
time, effort, cost
from idea to tryout

reliable realization is difficult
too much
and unpredictable
development
time, effort, cost
from tryout to realization

engineering is difficult
some new features
late relative to competition
too much
material and labor cost

observed causes

25 years of historical growth

lack of overview large amount of
detailed documentation

size and complexity
of realization

size and complexity
of organization

inherent complexity of
system and context

human and cultural factors
high level of expertise
conservatism

suspected more specific root causes

coupling (dependencies)
higher than needed

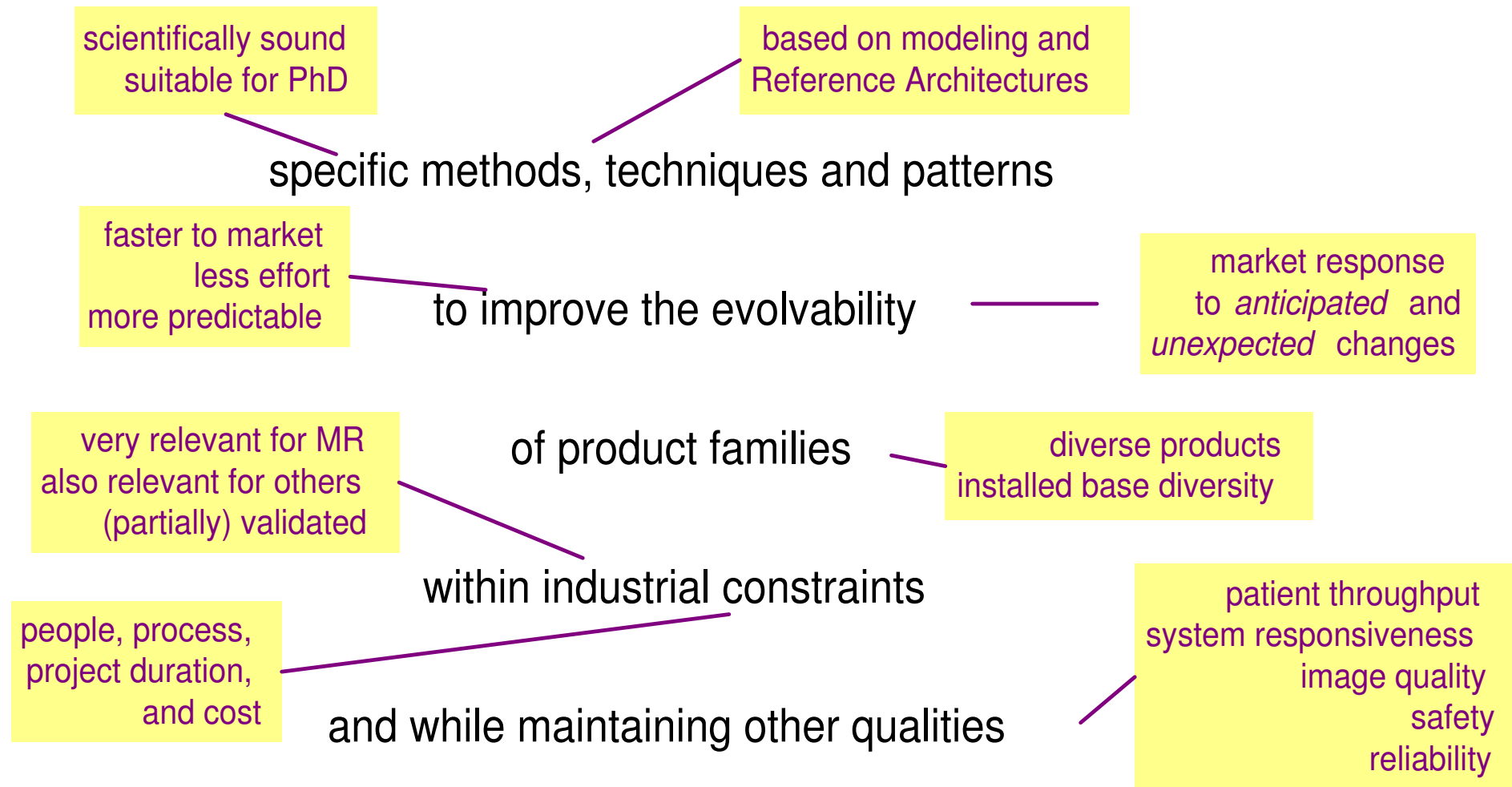
ineffective structure
(decomposition, interfaces)

insufficient
underpinning of decisions
by value and cost

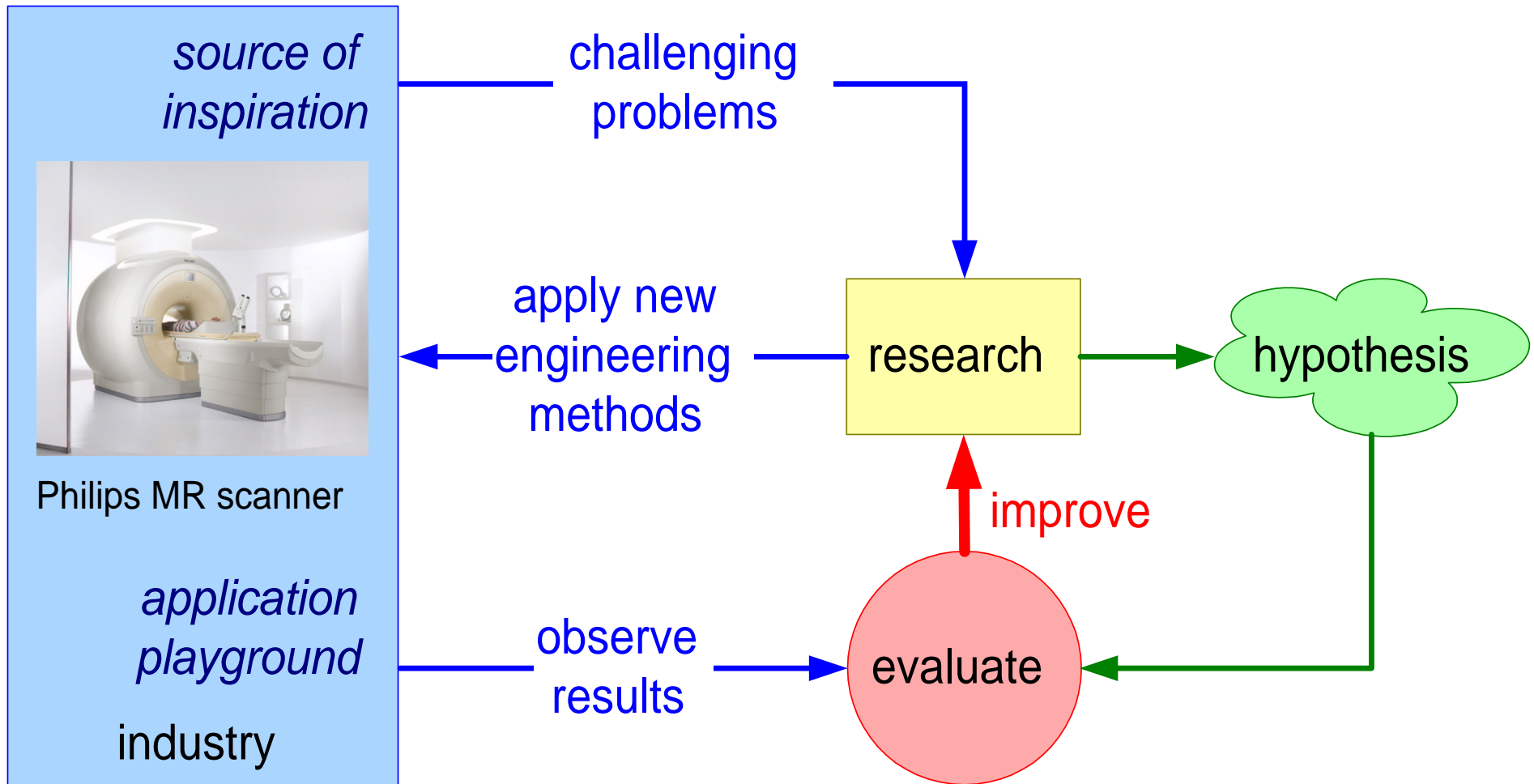
unbalance in
core/key/base

diversity of configurations

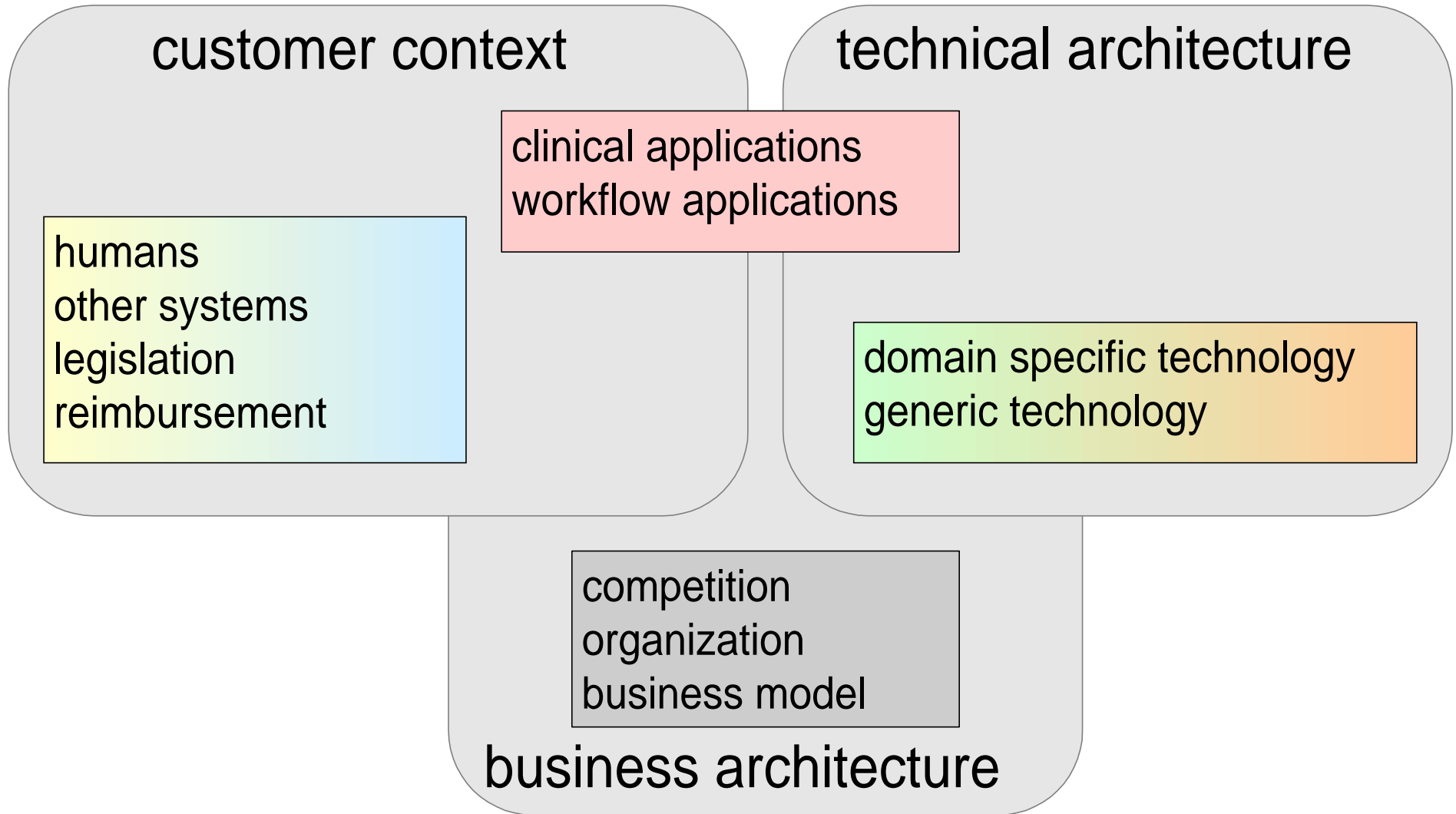
Darwin Project Goal



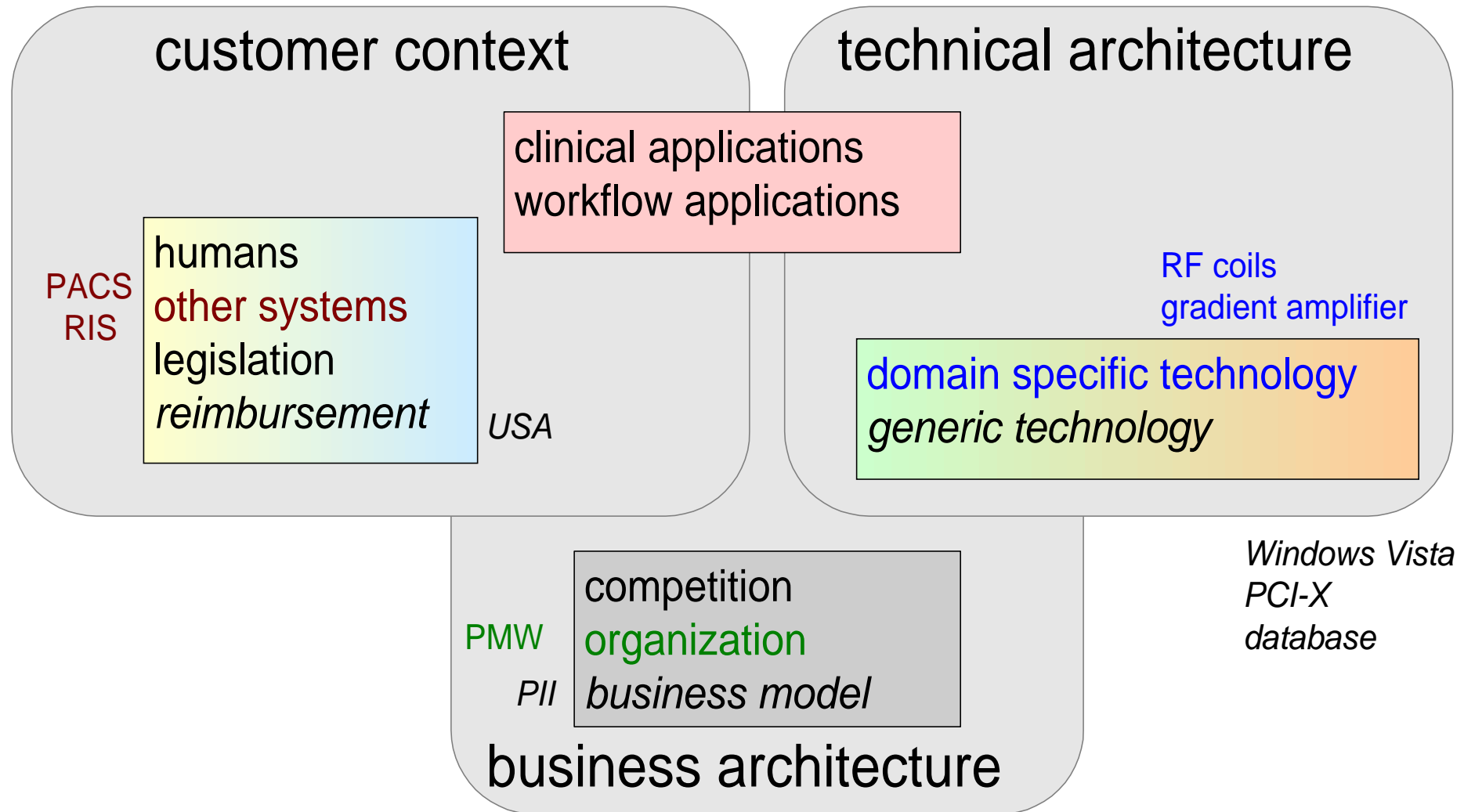
Darwin Research Model: Industry as Laboratory



Sources of Change



Sources of Change



Darwin Research Questions

How to transform into an evolvable product family architecture?

How to support decision making? *business wise*
technological

How to create overview? *by visualization*
by high-level modeling

How to mine the realization for implicit know how?

What are practical guidelines? *for decomposition*
for interface definition

What are patterns that support evolvability?

related research areas

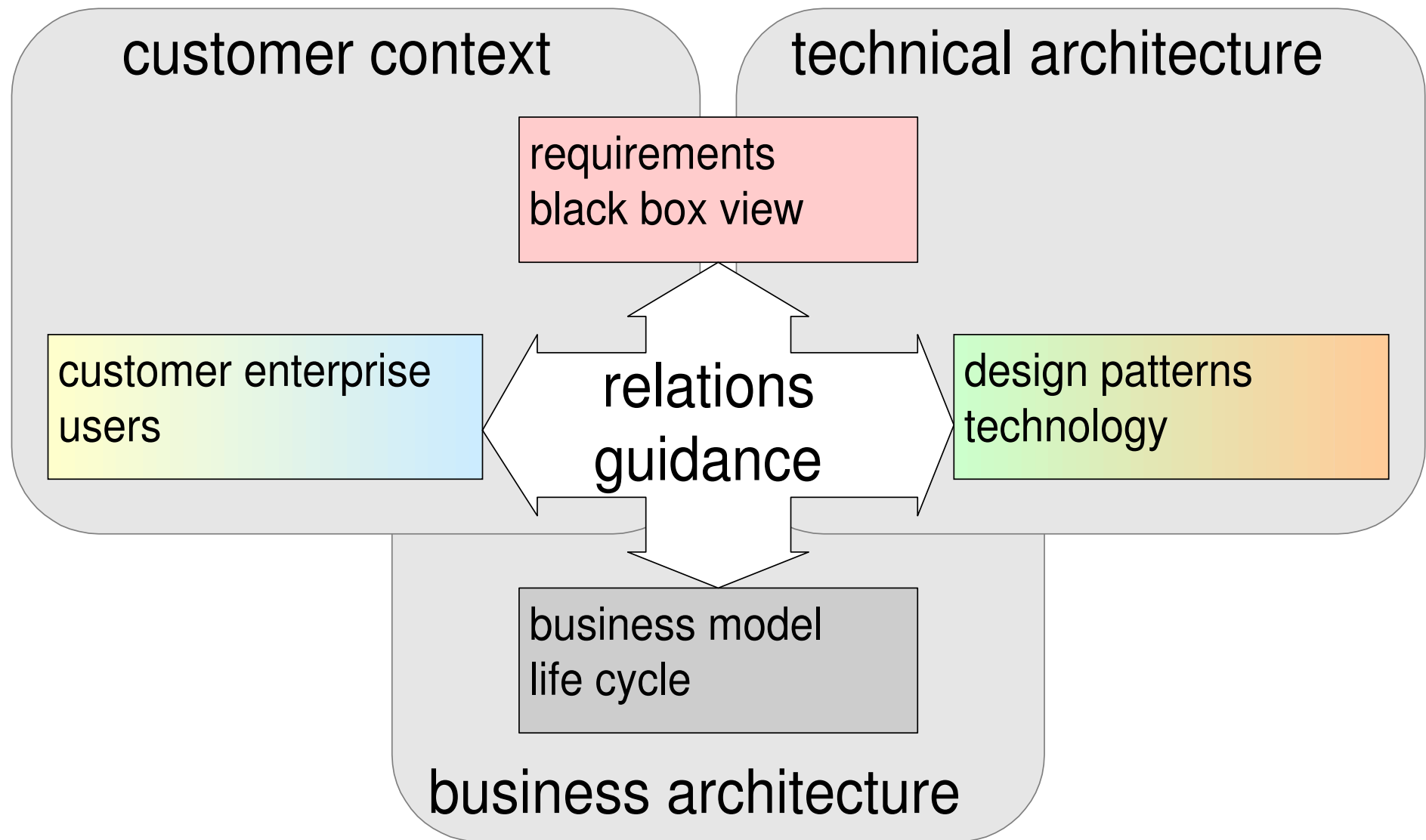
value analysis, e.g. real option
roadmapping

reference architecture
physical models, functional models,
budgeting, figures-of-merit,
state-diagrams, time-lines

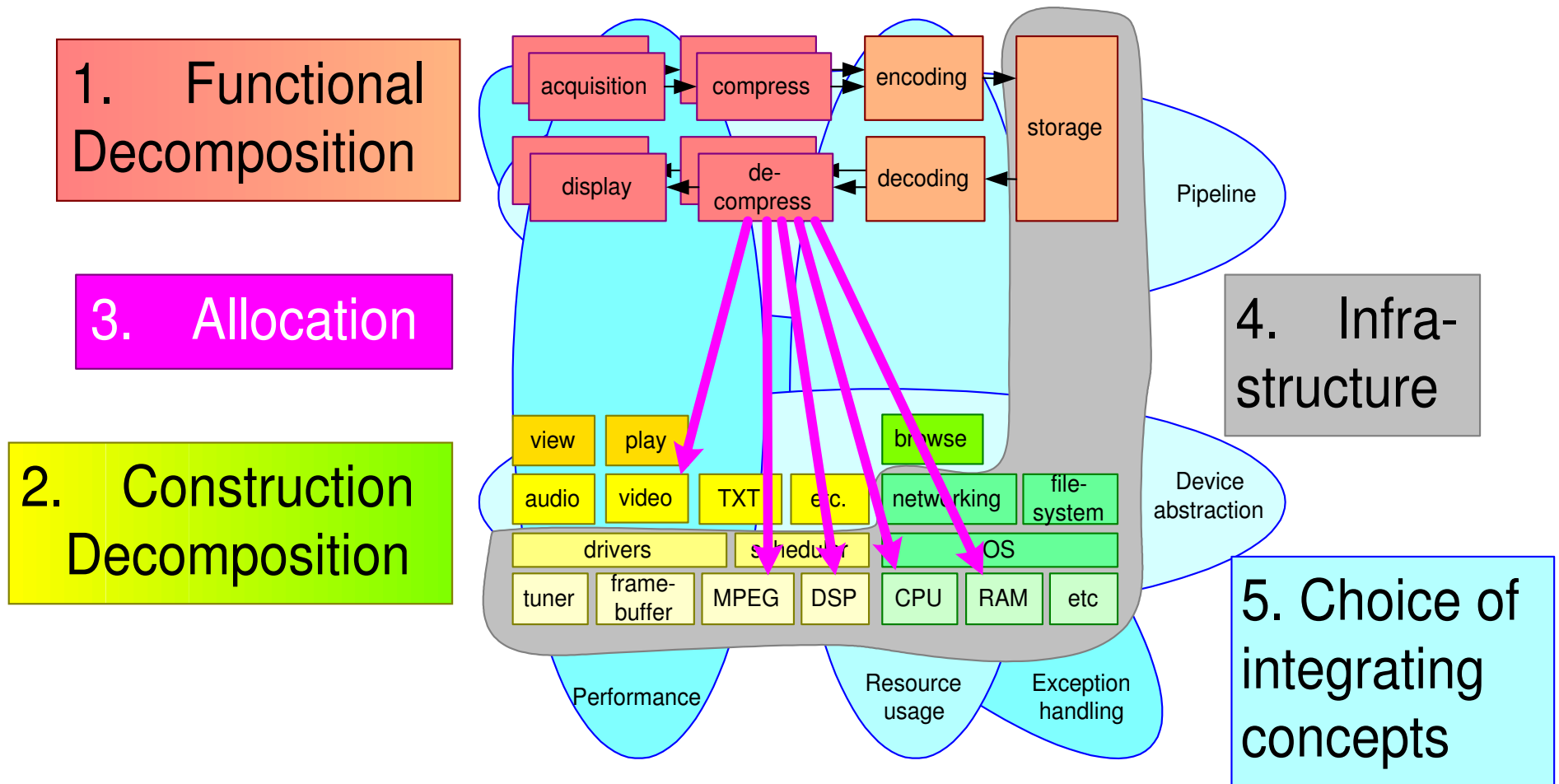
repository meta-data analysis
dynamic dependency analysis
semantic analysis

reference architecture
physical models, functional models,
qualities, behavior models
clustering, structure, set-based design

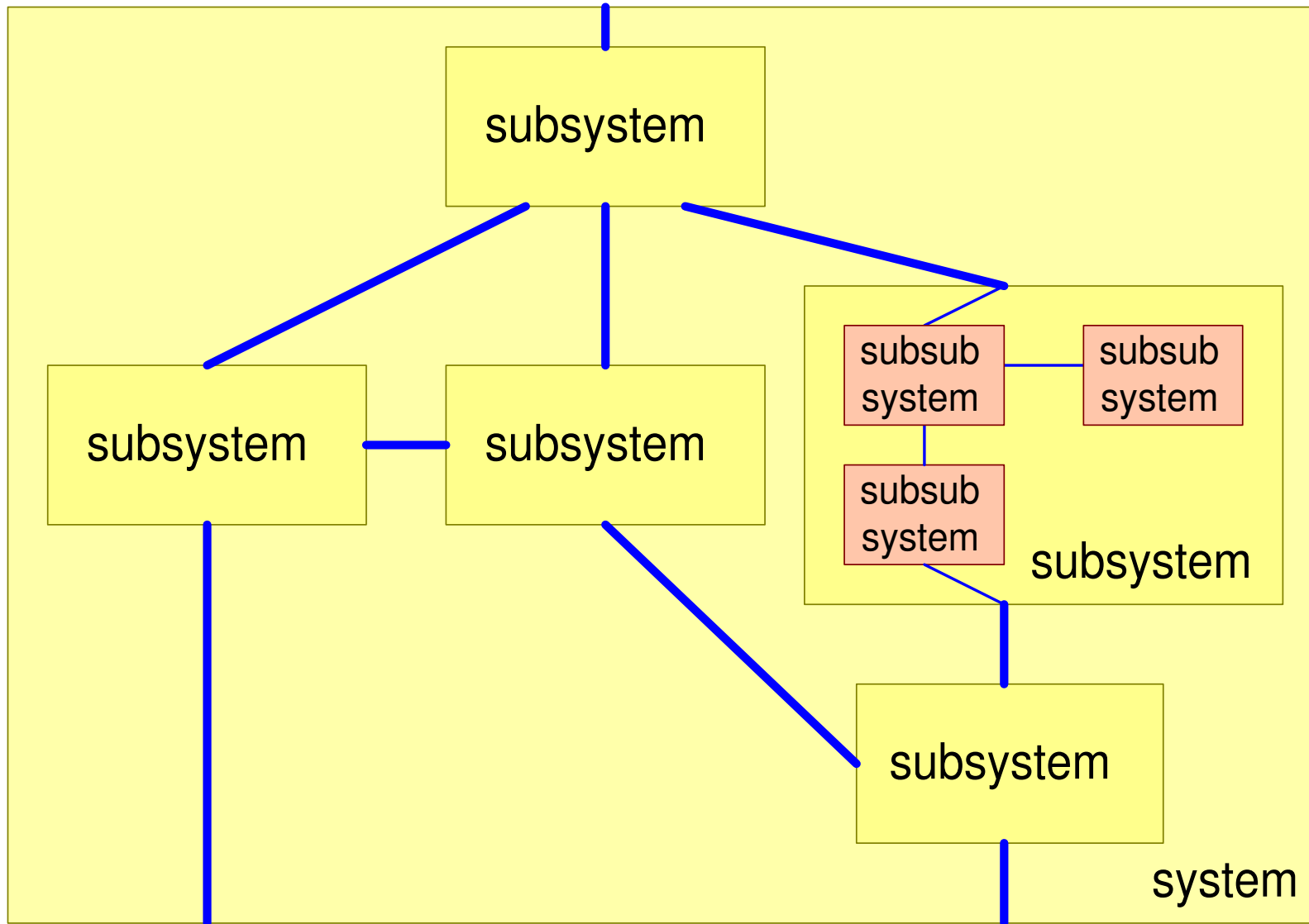
RA = Business Arch. + Technical Arch. + Customer Context



Technical Architecture

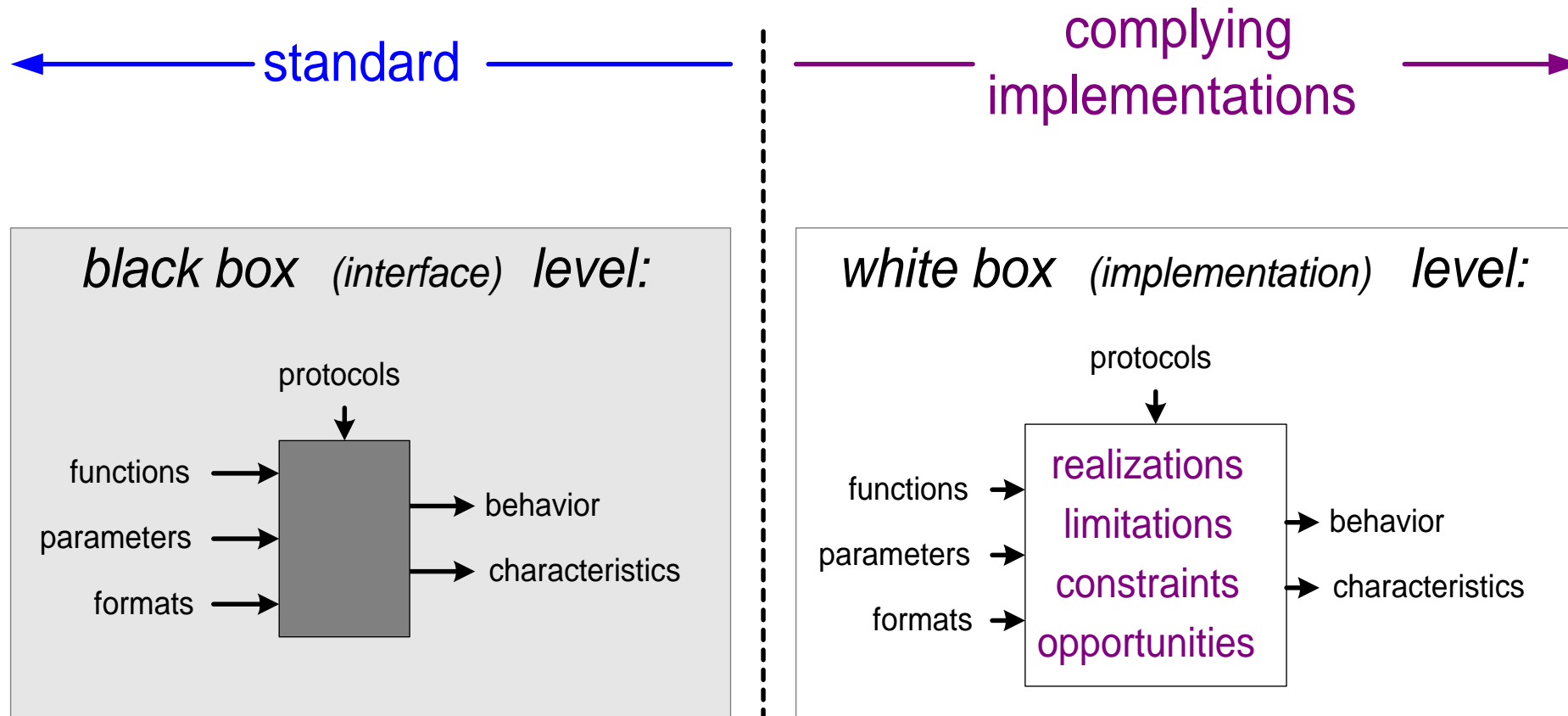


Decomposition and Interfaces

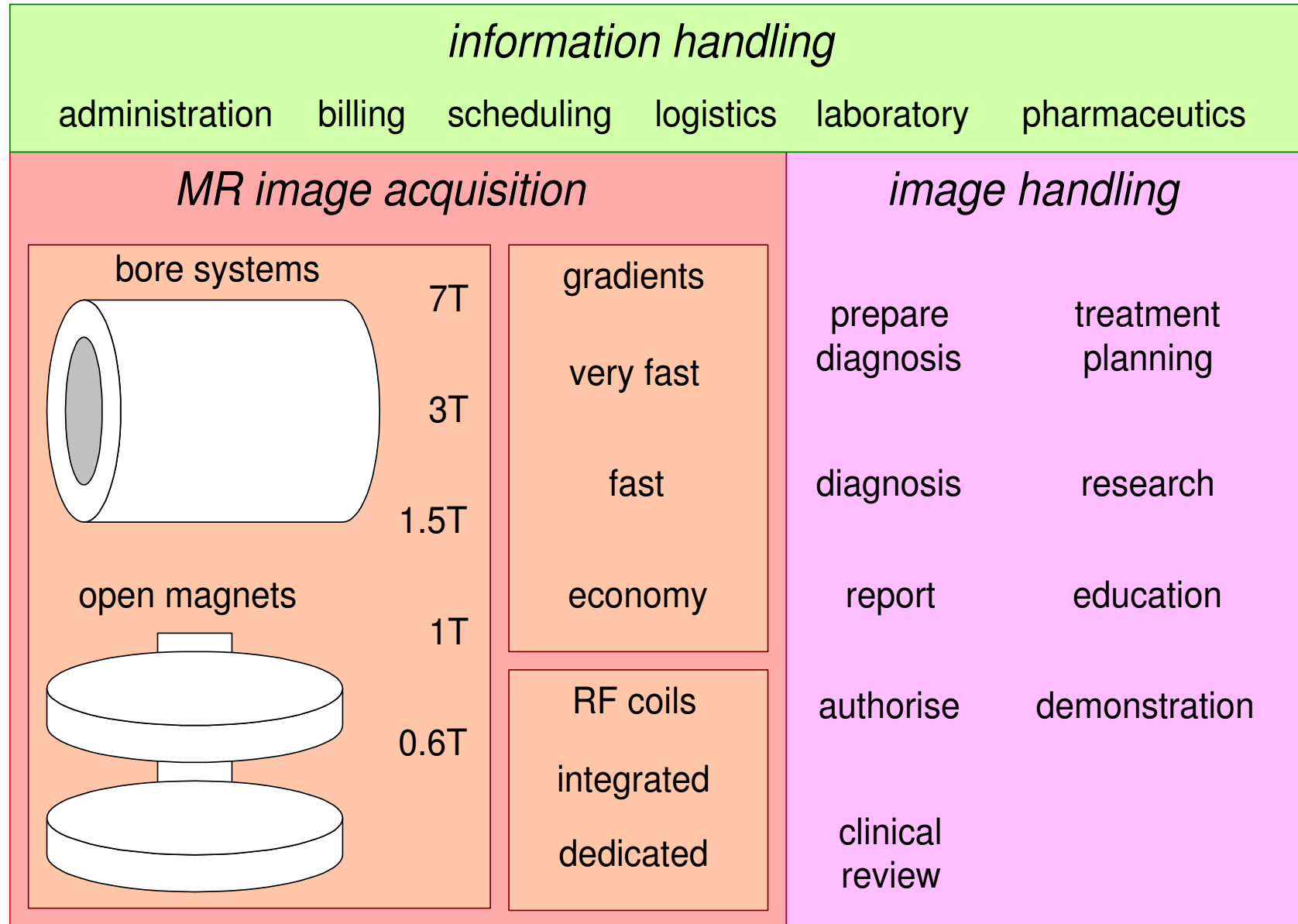


interface

Interface much more than functions + parameters

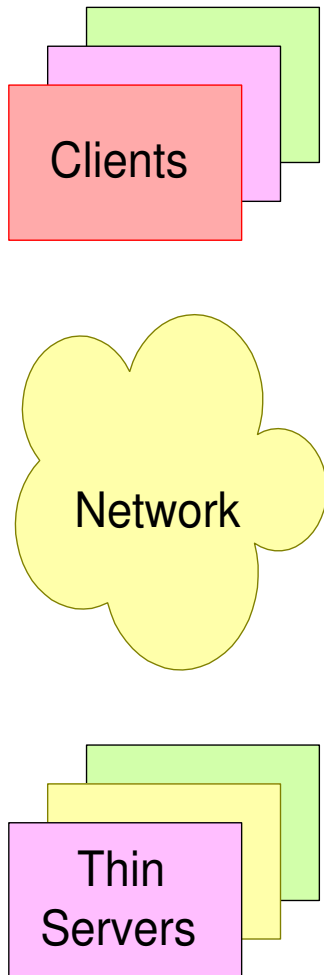


Integration and Diversity

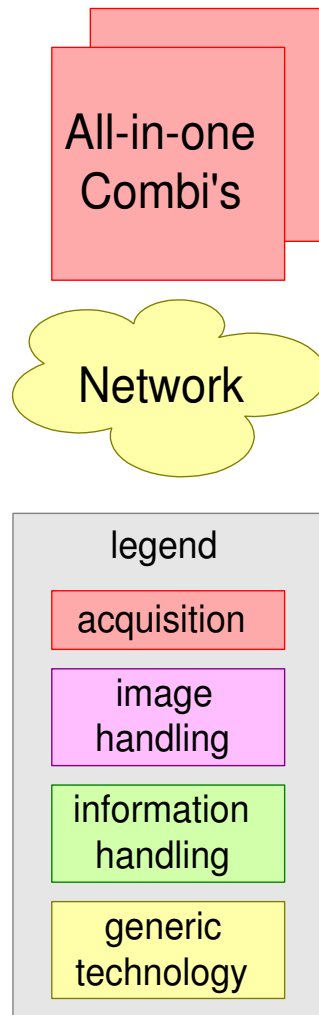


Distribution Scenario's

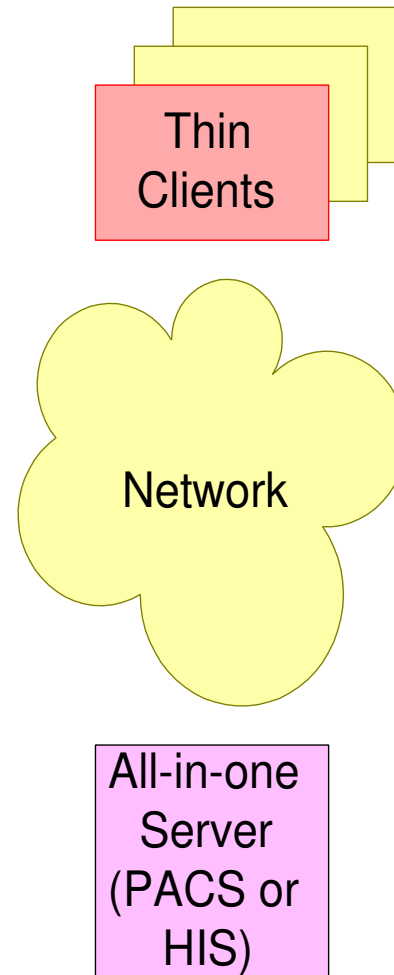
A "Thin Servers"



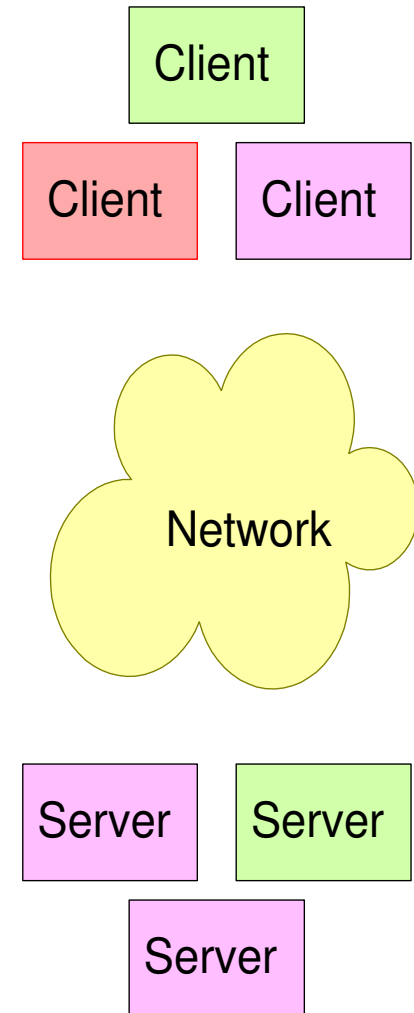
B "All-in-one" Combi's



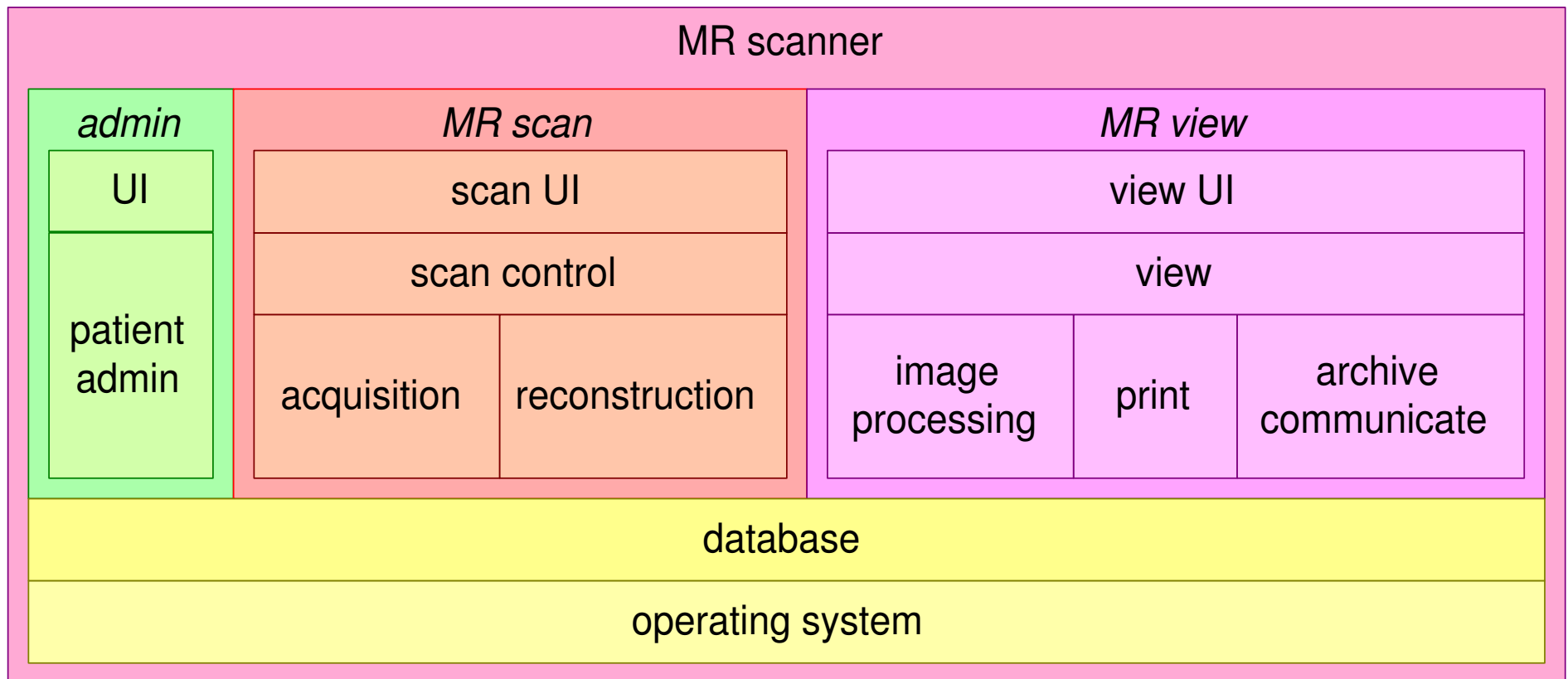
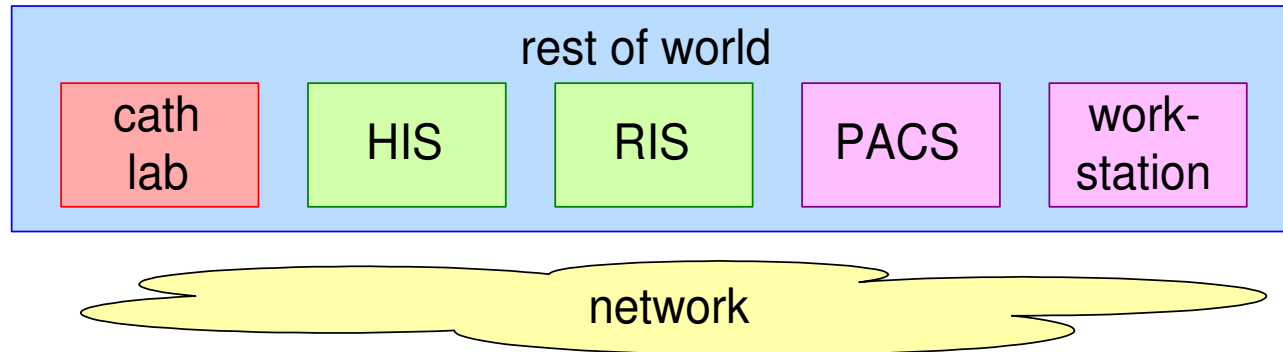
C "All-in-one" server



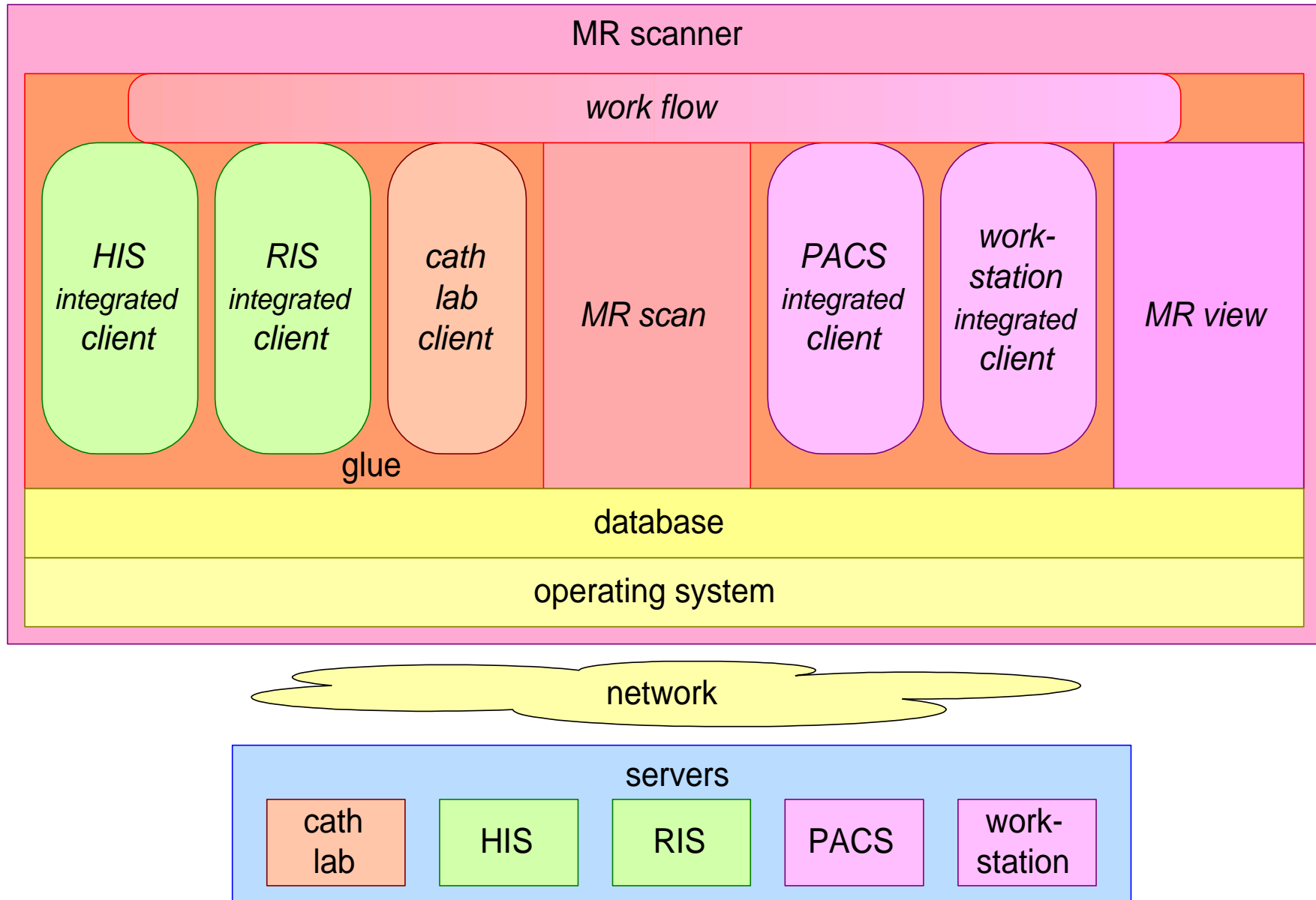
D "Modular"



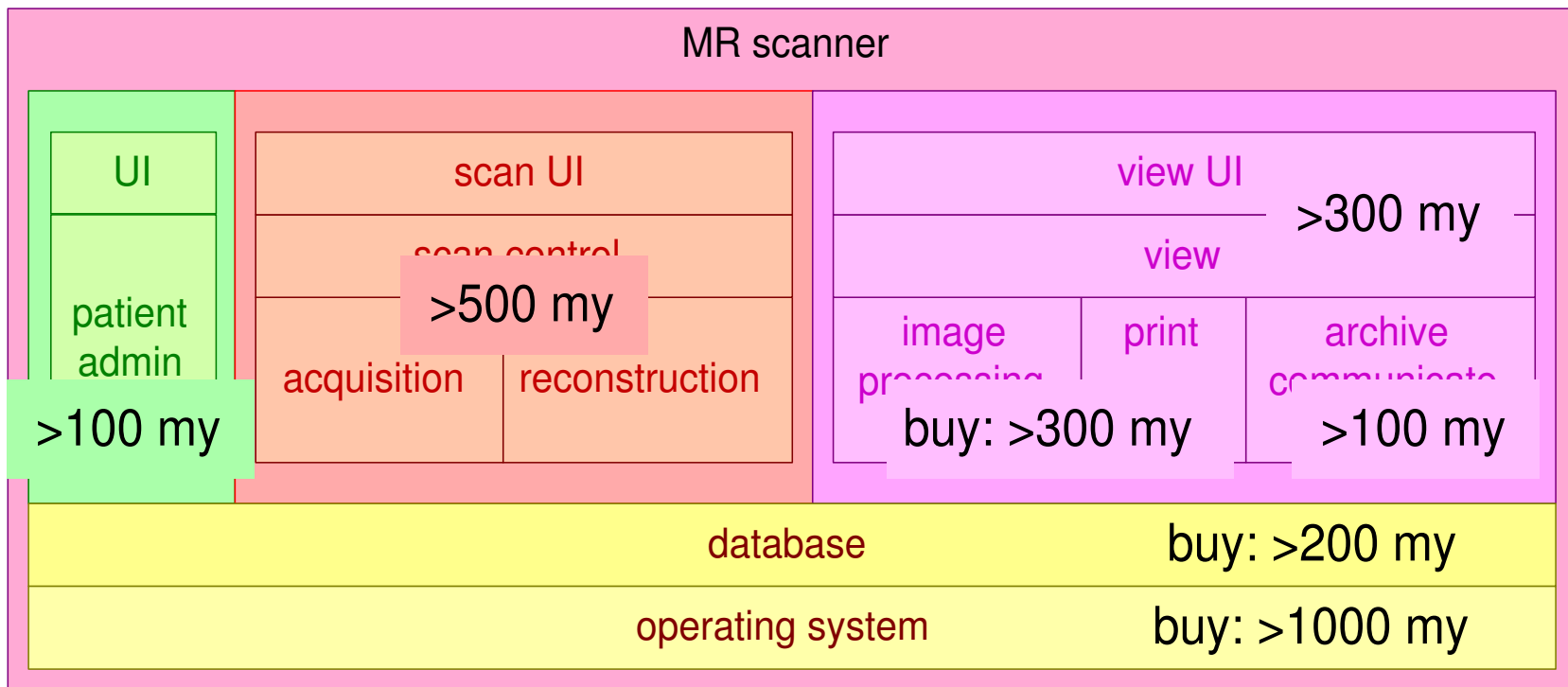
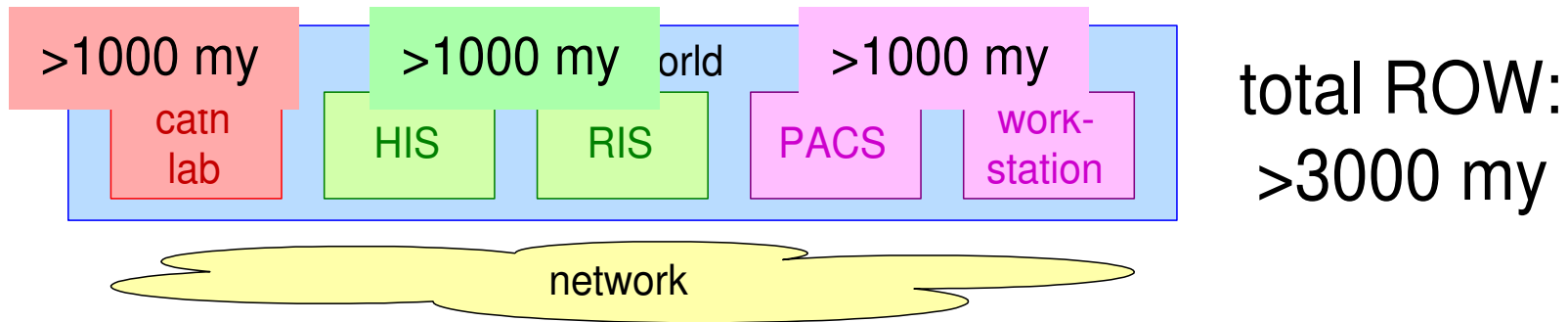
Simplistic Architecture



Future Simplistic Architecture



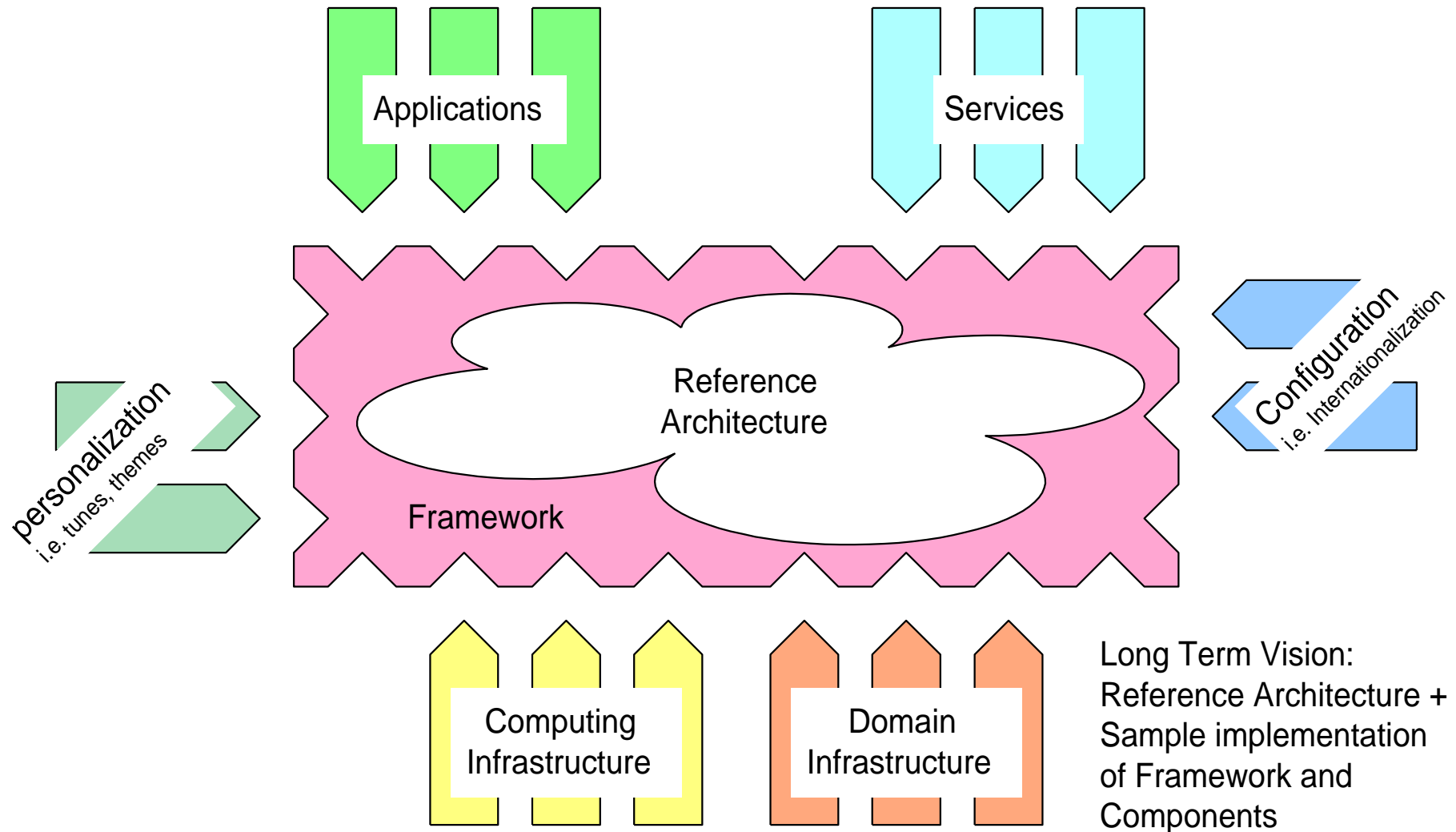
Available Code Assets



total make: >1000 my

total buy: >1500 my

Example Long Term Vision



Conclusion: Refactoring the Architecture is a must

